

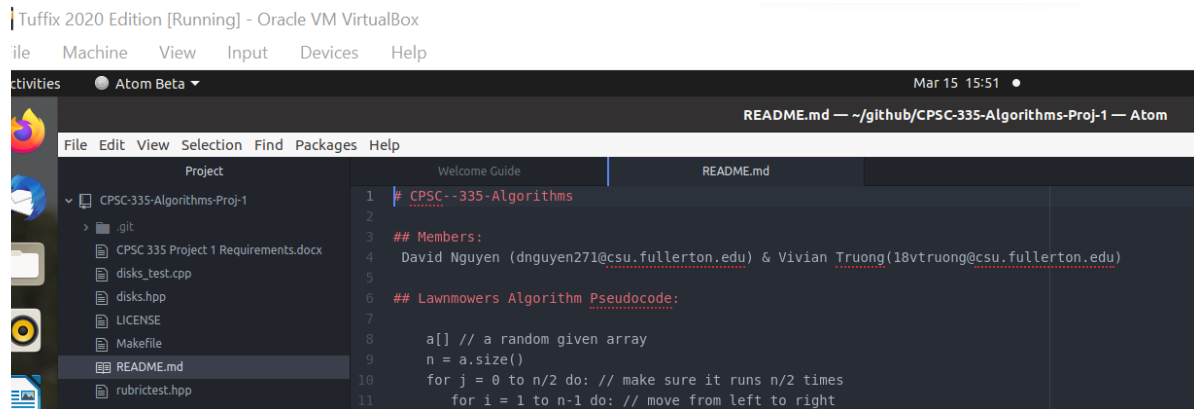
1.

David Nguyen (dnguyen271@csu.fullerton.edu)

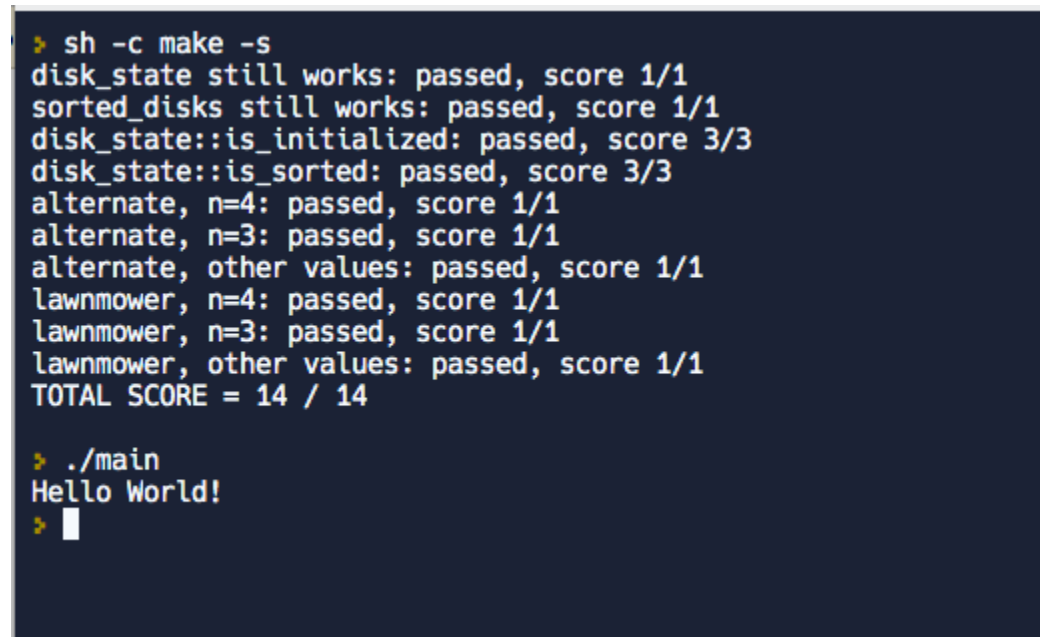
Vivian Truong(vtruong72@csu.fullerton.edu)

CPSC 335 Project 1

## 2. Screenshot inside Tuffix



## 3. Screenshot of execution



#### 4. Step count and efficiency

### Lawnmowers Algorithm Pseudocode:

```
a[] // a random given array

n = a.size()

for j = 0 to n/2 do: // make sure it runs n/2 times      n/2+1
times

    for i = 1 to n-1 do: // move from left to right      n-1
times

        if (a[i] == black && a[i+1] != black): // check for
swappable elements      3 tu

            swap;

//      S.C. =  $3(n^2-n)/2 + 3n-3$ 

    for j = n-1 down to 1 do: // move from right to left      n-1
times

        if(a[j] == white && a[j-1] != white): // check for
swappable elements      3 tu

            swap;

//      S.C. =  $2n-2$  so final step count  $3n^2-9n+12/2$  .
Time complexity is  $O(n^2)$ 
```

### Alternate Algorithm Pseudocode:

```
a[] // a random given array

n = a.size()
```

```

bool sorted

while(!sorted) do:

    for i = 1 to n-1 do: // move from left to right
n-1 times

        else if (a[i] == black && a[i+1] != black): // check for
swappable elements    3 tu

            swap;

//                                S.C.:  $3n-3$ 

    for i = 2 to n-2 do: // check the secondleft to secondright
disc        n-3 times

        else if (a[i] == black && a[i+1] != black): // check for
swappable elements    3 tu

            swap;

//                                S.C.:  $3n-9$  so  $9n^2-36n+27$  .

Time complexity is  $O(n^2)$ 

```

## 5. Time Complexity

We have concluded that both pseudocode algorithms are  $O(n^2)$  based on the leading terms.

### Lawnmowers:

LawnMower

$a[]$   
 $n = a.size()$   
 for  $j = 0$  to  $n/2$  do:  $\frac{n}{2} - 0 + 1 = \frac{n}{2} + 1$  times  
 for  $i = 1$  to  $n-1$  do:  $(n-1) - 1 + 1 \Rightarrow n-1$  times  
 if  $(a[i] == \text{black} \ \&\& \ a[i+1] != \text{black})$  : 3tu  
 swap

S.C.  $\frac{n}{2} + 1 * (n-1) * (3 + \max(0,0))$   
 $\frac{2+n}{2} (n-1) * 3$   
 $\rightarrow \frac{n^2}{2} - \frac{n}{2} + n - 1 * 3$   
 $\frac{n^2 - n}{2} (n-1) * 3 \Rightarrow \boxed{\frac{3(n^2 - n)}{2} + 3n - 3}$

for  $j = n-1$  down to 1 do  $\rightarrow \frac{1 - (n-1)}{-1} + 1 = n-2+1 \Rightarrow n-1$  times  
 if  $(a[j] == \text{white} \ \&\& \ a[j-1] != \text{white})$  : 3tu  
 swap

S.C.  $n-1 * 2 + \max(0,0)$   
 $2n-2$   
 Final S.C. :  $\boxed{\frac{3n^2 - 9n + 12}{12}}$

Time complexity:  $O(n^2)$



Alternate:

Alternate

```
q[]
n = q.size()
bool sorted
while (!sorted) do:
    for i = 1 to n-1 do: (n-1) - i + 1 [n-1] times
        else if (q[i] == black && q[i+1] != black) 3 + u
            swap
```

$(n-1) * (3 + \max(0, 0))$   
 $n-1 * 3$   
 $[3n-3]$

```
for i = 2 to n-2 do: (n-2) - i + 1 = n-3 times
    else if (q[i] == black && q[i+1] != black) 3 + u
        swap
```

$(n-3) * (3 + \max(0, 0))$   
 $n-3 * 3$   
 $3n-9$

Total s.c. =  $(3n-3)(3n-9)$   
 $9n^2 - 36n + 27$

Time complexity:  $O(n^2)$